Lab Report: Filtering Multimedia Signals Using an FIR Filter

Cover Page:

Lab Number: 5

Name: William Ewanchuk

Student ID: 1692036

Section: D51

Date Lab Performed: 12/5/2024

1. Introduction:

In this lab, we applied digital filters to multimedia signals, specifically audio and image signals, to analyze their frequency spectrum and remove noise or unwanted components. Digital filters, which modify the frequency content of a signal, are commonly used in both audio and image processing for tasks such as noise removal and signal enhancement. This report details the steps involved in applying high-pass, low-pass, and band-stop filters to an audio signal, as well as performing a low-pass filtering operation on a noisy image. The goal was to observe the effect of each filter on both the signal and the filtered output and make necessary adjustments to optimize noise removal.

2. Audio Signal Filtering:

2.1 Lowpass Filter Design (Q1):

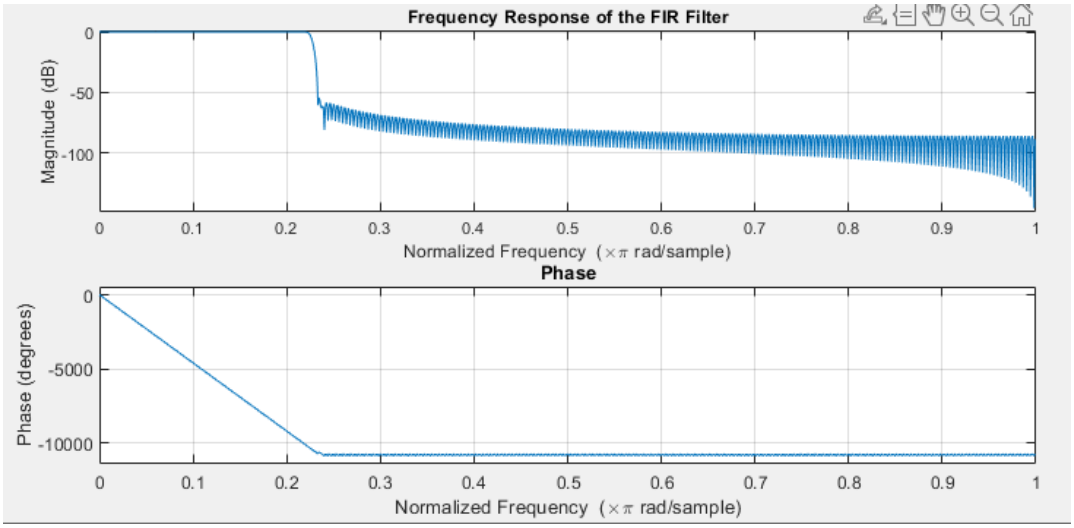
The first part of the lab involved designing a 513-tap lowpass FIR filter using MATLAB's fir1 function. The lowpass filter was designed with a cutoff frequency of 2500 Hz, which means frequencies above this value would be attenuated by the filter. The stopband ripples were restricted to-50 dB to ensure an adequate level of attenuation. The choice of a Hamming window was made to smooth the transition between the passband and stopband, ensuring that the signal was filtered effectively while maintaining a reasonable level of performance.

- Filter Coefficients Calculation: The MATLAB code used to generate the filter coefficients is as follows: see appendix for question 1 matlab code

2.2 Frequency Response of the Filter:

The frequency response of the lowpass filter was plotted using MATLAB's freqz function. This function computes and displays the magnitude and phase responses of the filter.

- Plot: Frequency Response of the Lowpass Filter:



The plot demonstrates that frequencies below the cutoff point (2500 Hz) are passed with minimal attenuation, while frequencies above this threshold are heavily attenuated, as expected from a lowpass filter.

2.3 Filtering the Audio Signal:

The next step was to filter the audio signal, which was read from the file love_mono22.wav. The filter was applied using MATLAB's filter function, and the resulting signal was stored in x_filtered.

- MATLAB Code for Filtering: see appendix for question 2 matlab code

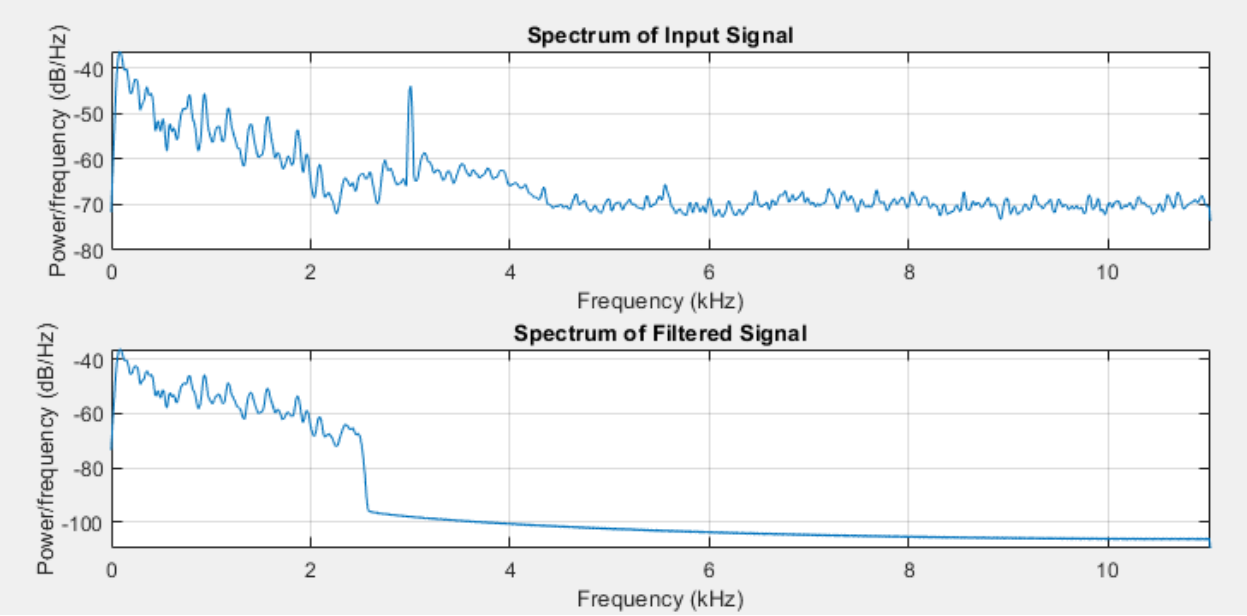
[x, Fs_audio] = audioread('love_mono22.wav'); % Read audio file

x_filtered = filter(filter_coeff, 1, x); % Apply filter to the audio signal

2.4 Spectrum Analysis of Original and Filtered Signals:

The pwelch function was used to compute the power spectral density (PSD) of both the original and filtered signals. The comparison between the two spectra provides insight into how the lowpass filter affected the frequency content of the signal.

- Plots: Spectrum of Original and Filtered Signals:



The power spectrum of the original signal shows a significant presence of high-frequency components that were effectively removed in the filtered signal. This indicates that the lowpass filter successfully attenuated the higher frequencies while preserving the lower-frequency components.

2.5 Comments on the Difference:

Listening to both the original and filtered signals reveals a noticeable reduction in high-frequency noise and distortion in the filtered signal. The clarity of the audio improved as the high-frequency components were attenuated, leaving behind the core audio signal with reduced noise.

---

3. Highpass Filter Design (Q2):

3.1 Highpass Filter Design:

For the second part of the lab, a 513-tap highpass FIR filter was designed with a cutoff frequency of 5000 Hz. This filter was intended to remove low-frequency components while preserving the higher frequencies above 5000 Hz. Again, a Hamming window was used to reduce the ripple in the frequency response.

- MATLAB Code for Highpass Filter:

wc = fc / (Fs / 2); % Normalized cutoff frequency

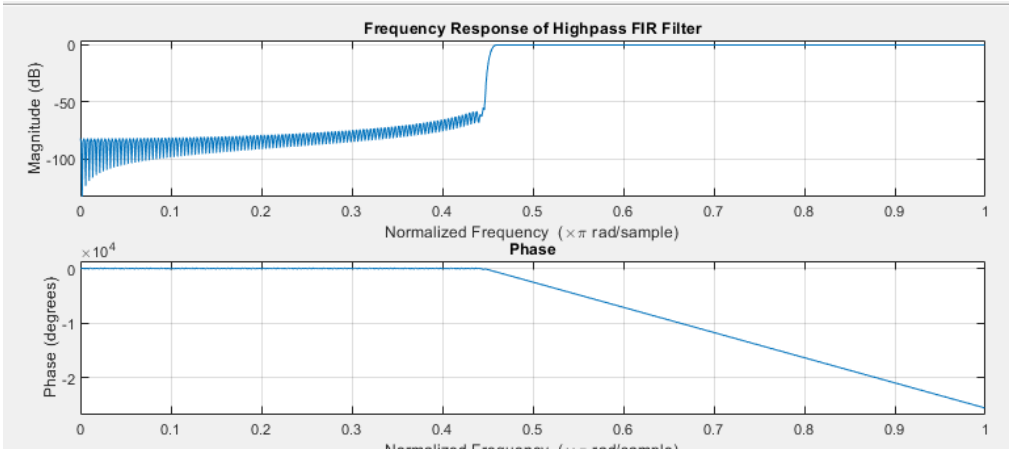
window = hamming(513); % Hamming window

filter_coeff = fir1(512, wc, 'high', window); % FIR highpass filter coefficients

3.2 Frequency Response of the Highpass Filter:

The frequency response of the highpass filter was plotted using the freqz function.

- Plot: Frequency Response of the Highpass Filter:



The plot shows that frequencies above 5000 Hz are passed with minimal attenuation, while frequencies below 5000 Hz are attenuated.

3.3 Filtering the Audio Signal:

The same audio file love_mono22.wav was filtered using the highpass filter designed above.

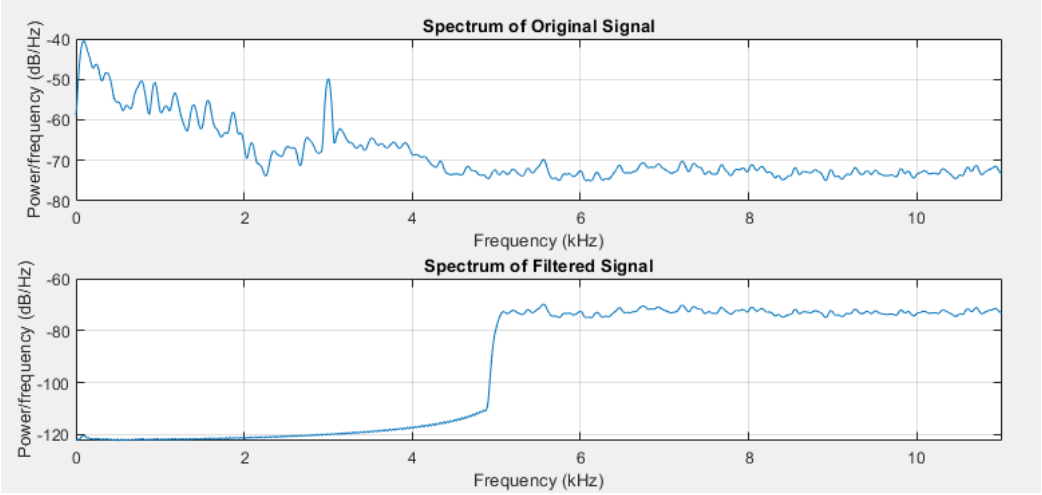
- MATLAB Code for Filtering: see appendix for question 2 matlab code

x_filtered = filter(filter_coeff, 1, x); % Apply highpass filter to the audio signal

3.4 Spectrum Analysis of Original and Filtered Signals:

As with the lowpass filter, the pwelch function was used to compute and compare the power spectra of the original and highpass-filtered signals.

- Plots: Spectrum of Original and Filtered Signals:



The comparison shows that the highpass filter has effectively removed the low-frequency components, leaving the high-frequency content intact.

3.5 Comments on the Difference:

The highpass filter successfully removed the low-frequency components, leaving only the higher-frequency elements in the signal. This can be heard in the audio playback, where the bass elements are attenuated, and the treble sounds are more prominent.

---

4. Noise Removal via Band-Stop Filtering (Q3):

4.1 Identifying Noise in the Spectrum:

In this section, the audio signal was analyzed for noise. Using the frequency spectrum of the noisy audio signal, we identified the frequency range where most of the noise's energy was located. Based on this observation, we designed a band-stop filter to remove this specific range of noise frequencies. The chosen cutoff filter frequencies were 2850 hz and 3150 hz.

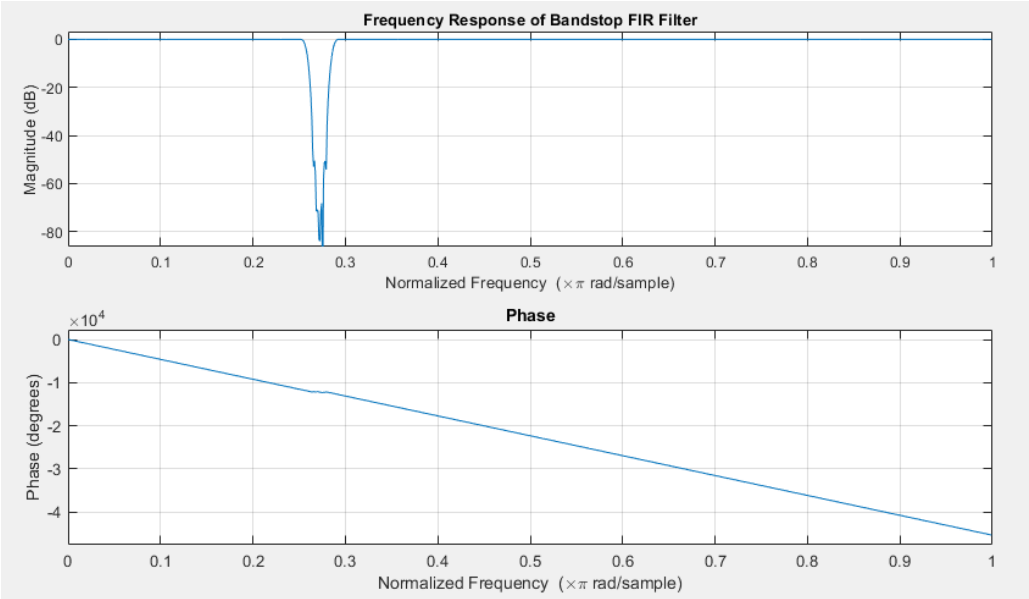
- Filter Design: The filter was designed using the fir1 function with an appropriate cutoff frequency range for the band-stop filter.
- MATLAB Code for Band-Stop Filter:

filter_coeff = fir1(512, [low_cutoff, high_cutoff], 'stop', window);

4.2 Frequency Response of the Band-Stop Filter:

The frequency response of the band-stop filter was plotted using freqz.

- Plot: Frequency Response of the Band-Stop Filter:



4.3 Filtering the Audio Signal:

The noisy audio signal was filtered using the band-stop filter.

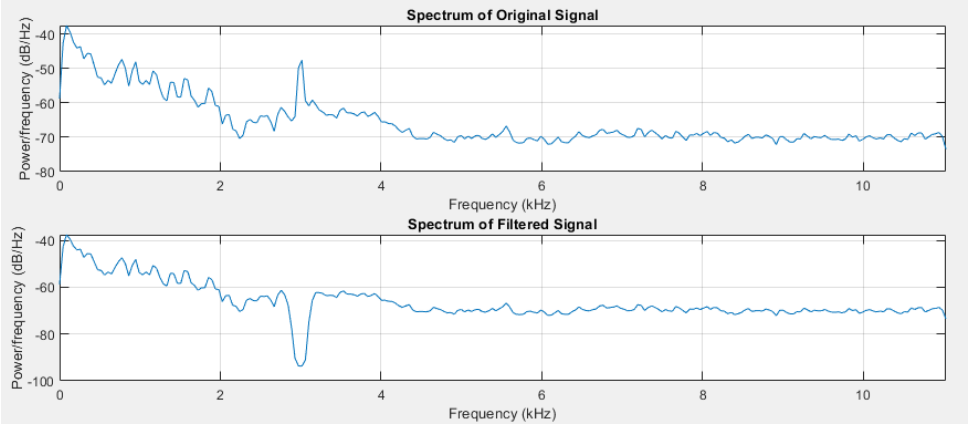
- MATLAB Code for Filtering: see appendix for question 3 matlab code

x_filtered = filter(filter_coeff, 1, x); % Apply band-stop filter to the audio signal

4.4 Spectrum Analysis of Original and Filtered Signals:

The pwelch function was again used to compare the power spectra of the original and filtered signals.

- Plots: Spectrum of Original and Filtered Signals:



The filtered signal shows a significant reduction in the noise frequencies compared to the original signal.

4.5 Comments on the Difference:

After applying the band-stop filter, the noise was effectively removed, and the audio signal became cleaner. The noise reduction is evident in the power spectrum, where the unwanted frequencies have been attenuated, and the signal's clarity has improved.

5. Image Filtering:

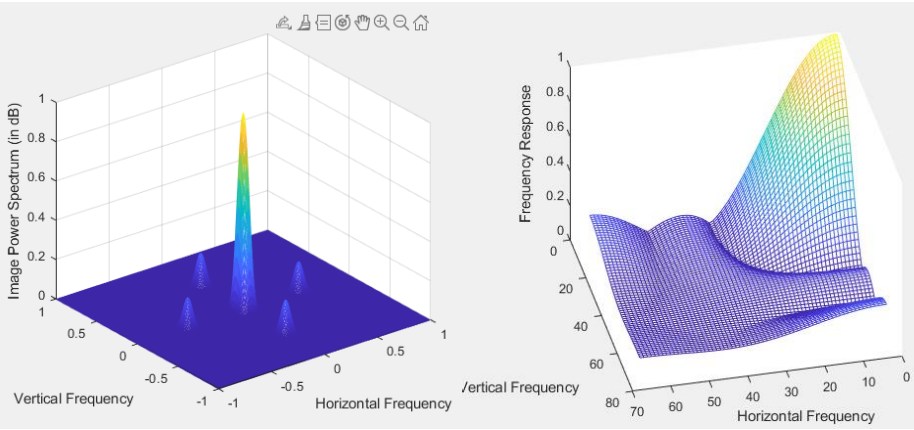
5.1 Filtering the Noisy Image:

For the final section of the lab, a low-pass filter was applied to an image corrupted by noise. The filter used was a 5x5 kernel, which served to smooth the image and remove the high-frequency noise.

5.2 Observations on Image Filtering:

After applying the filter, the image became smoother with reduced contrast between the hair and dark background. The grid-like noise pattern that appeared as a vertical and horizontal striping pattern was successfully removed.

- Plot: Noisy and Filtered Image:



5.3 Comments on the Image Filter:

The filtering operation reduced the sharp edges and contrast of the image. The noise, which was visible as grid-like patterns, was completely eliminated, and the image became more visually uniform. However, some fine details were lost in the darker colors due to the nature of the low pass filter applied to the signal. The distinguishing features surrounding the hair and the background was lost due to the filtering.

6. Conclusion:

In this lab, we applied various FIR filters to both audio and image signals. We designed and implemented low-pass, high-pass, and band-stop filters for audio signal processing to remove unwanted frequency components and noise. For the image processing task, a low-pass filter was applied to reduce high-frequency noise. In both cases, the filters successfully removed unwanted

noise and improved the quality of the signals. The lab demonstrated the practical application of FIR filters in real-world multimedia signal processing tasks.

7. Appendix: Matlab code:

Question 1 low pass filter design with comparison of before and after filtering subplots:

```
% Parameters for filter design
Fs = 22050;                 % Sampling frequency in Hz
fc = 2500;                  % Cut-off frequency in Hz
wc = fc / (Fs / 2);         % Normalized cut-off frequency
N = 513;                    % Filter order (number of taps = N)
window = hamming(N);        % Hamming window
filter_coeff = fir1(N-1, wc, 'low', window); % Low-pass FIR filter design

% Plot the frequency response of the filter
figure;
freqz(filter_coeff, 1, 1024);  % Frequency response
title('Frequency Response of the FIR Filter');

% Read the audio signal
[x, Fs_audio] = audioread('love_mono22.wav'); % Load audio signal
if Fs_audio ~= Fs
    error('Sampling frequency mismatch: Expected %d Hz, but got %d Hz', Fs, Fs_audio);
end

% Filter the audio signal
x_filtered = filter(filter_coeff, 1, x); % Filter the audio signal

% Compare the spectrum of the input and output signals
window_size = 1024;
overlap = window_size / 2;
nfft = 2048;

% Plot comparison in two subplots
figure;
subplot(2, 1, 1); % First subplot for input signal spectrum
pwelch(x, window_size, [], nfft, Fs);
grid on;

title('Spectrum of Input Signal');

subplot(2, 1, 2); % Second subplot for filtered signal spectrum
pwelch(x_filtered, window_size, [], nfft, Fs);
grid on;
title('Spectrum of Filtered Signal');

% Save the modified output signal
audiowrite('filtered_love_mono22.wav', x_filtered, Fs); % Save output

% Play both signals
disp('Playing original signal...');
%sound(x, Fs);  % Play original signal
%pause(length(x) / Fs + 1); % Wait for playback to complete

disp('Playing filtered signal...');
%sound(x_filtered, Fs); % Play filtered signal

disp('Script completed! Listen to the difference between the original and filtered signals.');
```

Question 2 matlab code Left figure and Question 3 matlab code question 4

```
% Highpass FIR filter design
filter_coeff = fir1(N-1, wc, 'high', window);

% Plot frequency response
figure;
freqz(filter_coeff, 1, 1024); % Frequency response
title('Frequency Response of Highpass FIR Filter');

% Read the audio signal
[x, Fs] = audioread('love_mono22.wav'); % Load the audio file
x = x(:, 1); % Ensure it's a single channel (mono)

% Filter the audio signal
x_filtered = filter(filter_coeff, 1, x);

% Calculate and compare spectra
figure;
subplot(2, 1, 1);
pwelch(x, window, [],f, Fs); % Spectrum of the original signal
title('Spectrum of Original Signal');
subplot(2, 1, 2);
pwelch(x_filtered,window, [], f, Fs); % Spectrum of the filtered signal
title('Spectrum of Filtered Signal');

% Save the filtered signal as a .wav file
audiowrite('love_highpass_filtered.wav', x_filtered, Fs);

% Play the original and filtered signals (optional)
%sound(x, Fs); % Play the original signal
%pause(length(x)/Fs + 1); % Wait for the original signal to finish
%sound(x_filtered, Fs); % Play the filtered signal

% Comments on the difference
disp('Listen to the original and filtered signals to observe the attenuat
```

```
% MATLAB Script for Bandstop FIR Filter Design (2.95 kHz to 3.05 kHz)

% Parameters
Fs = 22050; % Sampling frequency in Hz
fc_low = 2850; % Low cutoff frequency in Hz (2.95 kHz)
fc_high = 3150; % High cutoff frequency in Hz (3.05 kHz)
N = 512; % Filter order (number of taps), you can increase this for sharper response

% Normalized cutoff frequencies (scaled by Nyquist frequency)
wc_low = fc_low / (Fs / 2);
wc_high = fc_high / (Fs / 2);

% Truncation window function (Hamming)
window = hamming(N);

% Design the bandstop filter using fir1
filter_coeff = fir1(N-1, [wc_low wc_high], 'stop');

% Plot frequency response
figure;
freqz(filter_coeff, 1, 1024); % Frequency response
title('Frequency Response of Bandstop FIR Filter');

% Read the audio signal
[x, Fs] = audioread('love_mono22.wav'); % Load the audio file
x = x(:, 1); % Ensure it's a single channel (mono)

% Filter the audio signal
x_filtered = filter(filter_coeff, 1, x);

% Calculate and compare spectra (in dB and kHz)
figure;
subplot(2, 1, 1);
pwelch(x, window, [], [], Fs); % Spectrum of the original signal
title('Spectrum of Original Signal');


subplot(2, 1, 2);
pwelch(x_filtered, window, [], [], Fs); % Spectrum of the filtered signal
title('Spectrum of Filtered Signal');


% Save the filtered signal as a .wav file
audiowrite('love_bandstop_filtered.wav', x_filtered, Fs);

% Play the original and filtered signals (optional)
%sound(x, Fs); % Play the original signal
%pause(length(x)/Fs + 1); % Wait for the original signal to finish
sound(x_filtered, Fs); % Play the filtered signal

% Comments on the difference
disp('Listen to the original and filtered signals to observe the bandstop behavior.');
```